# NAVAL POSTGRADUATE SCHOOL
## Monterey, California

# THESIS



THE USE OF A UNIX-BASED WORKSTATION IN
THE INFORMATION SYSTEMS LABORATORY

by

Charlotte V. Smith

March 1989

Thesis Advisor: Norman F. Schneidewind
Co-Advisor: Magdi Kamel

Approved for public release; distribution is unlimited.

LASSFIED
RITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| EPORT SECURITY CLASSIFICATION: UNCLASSFIED | 1b RESTRICTIVE MARKINGS |
| ECURITY CLASSIFICATION AUTHORITY | 3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited. |
| ECLASSIFICATION/DOWNGRADING SCHEDULE | |
| RFORMING ORGANIZATION REPORT NUMBER(S) | 5 MONITORING ORGANIZATION REPORT NUMBER(S) |
| IAME OF PERFORMING ORGANIZATION: val Postgraduate School — 6b OFFICE SYMBOL (If applicable): Code 54Ss | 7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School |
| DDRESS (City, State, and ZIP Code): nterey, CA 93943-5000 | 7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000 |
| IAME OF FUNDING/SPONSORING ORGANIZATION — 8b OFFICE SYMBOL (If applicable) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| DDRESS (City, State, and ZIP Code) | 10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO |

TITLE (Include Security Classification)
E USE OF A UNIX-BASED WORKSTATION IN THE INFORMATION SYSTEMS LABORATORY

PERSONAL AUTHOR(S)
ith, Charlotte V.

| TYPE OF REPORT | 13b TIME COVERED | 14 DATE OF REPORT (Year, Month, Day) | 15 PAGE COUNT |
|---|---|---|---|
| ster's Thesis | FROM ______ TO ______ | March 1989 | 78 |

SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author id do not reflect the official policy or postion of the Department of Defense the U.S. Government.

| COSATI CODES | | | 18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | UNIX; User Manuals |
| | | | |
| | | | |


ABSTRACT (Continue on reverse if necessary and identify by block number)
e Information Systems laboratory of the Department of Administrative Sciences s installed a UNIX-based workstation for student use. The UNIX operating stem, one of the most popular systems available today, is an attempt to ovide a powerful operating system that is largely machine-independent.

is thesis examines the capabilities and limitations of the UNIX operating stem as it pertains to the Information Systems student. It also provides a ser's manual for the students, thus, allowing them to gain a working knowledge the system and prepare them to take advantage of its capabilities.

e research concludes that, with proper administration, the use of UNIX rkstations can be very valuable to the Information Systems student.


| | |
|---|---|
| DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS | 21 ABSTRACT SECURITY CLASSIFICATION: Unclassified |
| a. NAME OF RESPONSIBLE INDIVIDUAL: rofessor Norman F. Schneidewind | 22b TELEPHONE (Include Area Code): (408) 646-2719 — 22c OFFICE SYMBOL: 54Ss |

D FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE

☆ U.S. Government Printing Office: 1986—606-243



UNCLASSIFIED

# The Use of a UNIX-based Workstation in the Information Systems Laboratory

by

Charlotte V. Smith
Lieutenant, United States Navy
B.A, Susquehanna University, 1981

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March 1989

## ABSTRACT

The Information Systems laboratory of the Department of Administrative Sciences has installed a UNIX-based workstation for student use. The UNIX operating system, one of the most popular systems available today, is an attempt to provide a powerful operating system that is largely machine-independent.

This thesis examines the capabilities and limitations of the UNIX operating system as it pertains to the Information Systems student. It also provides a user´s manual for the students, thus, allowing them to gain a working knowledge of the system and prepare them to take advantage of its capabilities.

The research concludes that, with proper administration, the use of UNIX workstations can be very valuable to the Information Systems student.

## TABLE OF CONTENTS

# I. INTRODUCTION

## A. BACKGROUND

The use of computers continues to grow exponentially in the world today. Operating systems, the master programs that control the computer's basic functions, are numerous and very diverse. The UNIX[1] operating system, one of the most popular systems available today, is an attempt to provide a powerful operating system that is largely machine-independent.

The capabilities of UNIX are vast. This thesis will research the UNIX operating system and provide a user's manual for Information Systems students at the Naval Postgraduate School.

## B. OBJECTIVES

The main objective of this thesis is to provide students with a basic understanding of the UNIX operating system, its functionality and its limitations. Appendix A to this thesis is a user's manual for the Information Systems laboratory's Symmetric 375 UNIX workstation. The workstation is a series 32000 microcomputer running the SYMMETRIX operating system, a version of the Berkeley 4.2 BSD UNIX operating system [Ref. 1:p. 15]. The manual

---

[1] UNIX is a trade mark of Bell Laboratories, Incorporated.

provides the student with documentation to successfully use the workstation. By providing this, Information Systems students will be able to gain a working knowledge of UNIX and will be prepared to take advantage of its capabilities.

This thesis also provides recommendations for the administration and management of the UNIX workstation. Procedures are suggested to ensure efficient operations while maintaining an appropriate level of security.

## C. METHODOLOGY

There were four phases to this research. The first phase consisted of an indepth study of the UNIX operating system. Phase two concentrated on learning to use the Symmetric 375 UNIX workstation. Phase three examined the possible uses of this system, its functionality to the students, and its recommended administration. Finally, the last phase, phase four, was the drafting of a user´s manual for the UNIX workstation.

## D. ORGANIZATION

This chapter serves as an introduction to the thesis and presents the objectives of the research. Chapter II examines the capabilities and limitations of the UNIX operating system. The conclusions of the research and the resulting recommendations are presented in Chapter III. These recommendations include how to manage the use of a UNIX workstation in the Information Systems laboratory.

Appendix A is the user´s manual which was developed for the Symmetric 375 workstation. This manual is intended to assist the user during the actual operation of the system. It can be used as a reference for the first time user, providing simple explanations of the log on sequence and initial use. It can also be valuable to the more experienced user, providing instructions on various commands and utilities. Appendix B provides reference information for the system administrator.

# II. CAPABILITIES AND LIMITATIONS

## A. INTRODUCTION

UNIX was first developed in 1969 by a programmer at AT&T, Ken Thompson. Thompson had written a program called "Space Travel" but was frustrated and disappointed because it was so expensive to run on the mainframe under its current operating system. Using a Digital Equipment Corporation (DEC) PDP-7 mini computer, Thompson proceeded to develop an operating system that would meet his needs. [Ref. 2:p. 7]

Thompson was soon joined by a fellow AT&T programmer named Dennis Ritchie and was eventually assisted by a few others from his group at AT&T. As UNIX evolved, its use spread greatly within the company and eventually out to universities. [Ref. 3:p. 464] The two main attributes which characterized the development of UNIX were the fact that it was developed by an extremely small team of people (mostly just two) and that it was developed by the users themselves and designed to fit their specific need. This method of origin and the systems distribution are responsible for the uniqueness of the system. It has lead not only to some great advantages because of existing capabilities, but it has also resulted in some definite limitations.

## B. CAPABILITIES

The UNIX system offers a wide variety of capabilities but the characteristics emphasized here will be those which have set it apart.

### 1. Hardware Environment

Although UNIX was initially developed on a DEC minicomputer, it is capable of running on a vast selection of computer hardware. This capability exist because, "...unlike other operating systems, it is largely machine-independent; this means that the UNIX system can run on mainframe computers as well as microcomputers and minicomputers." [Ref. 4:p. 1-2] With the availability of this kind of generic software, a new individual operating system no longer needs to be written each time a new computer hardware system is developed.

UNIX is sometimes referred to as "portable" because applications developed in UNIX on one computer system can be moved to another system which is running UNIX. Differences do exist between some versions of UNIX ; but, if some modifications are required to run the program on a new system, these should not be significant changes.

### 2. Processing Environment

The UNIX processing environment is interactive with multi-tasking and multi-user capabilities. Multi-tasking refers to the fact that many programs can be run concurrently. This capability gives UNIX the ability to

maximize resource utilization, increasing the efficiency of the computer and the productivity of the programmer. For example, multi-tasking permits the user to submit one file for compiling or spell checking while still editing another.

Multi-user is an environment where several users have access to the computer resources at the same time. By logging in individually to the system, users can be identified uniquely and be permitted to operate without interference from the other users of the system. They can also share programs and data if desired.

### 3. Filesystem

One of the outstanding features of UNIX is its simple filesystem. Using directories, the filesystem is organized in a hierarchical structure. This structure allows the user to create directories and organize files. Providing this ability enables UNIX to support the considerable number of files in a large system while still permitting the user to keep track of his files and the system´s organization.

### 4. Input/output

A very convenient capability available to UNIX users is input and output (I/O) redirection. Because (similar to the operating system itself) UNIX I/O is device-independent, the output of a process can be redirected to any appropriate device, such as the terminal, a printer, or even another file. The input of a process can also be specified; instead

of receiving input from the terminal, a process can be directed to accept it from another device or another process. By directing the output from one process to be the input to another, UNIX gives the user the ability to "pipe" several processes together.

### 5. Utilities and Operating Services

By design, UNIX is an operating system tailored for programmers [Ref. 5:p. 8]. It is suited ideally for software development. The UNIX system contains several hundred utilities which perform simple specific functions. A string of these available utilities can be piped together to create a personalized program which accomplishes a complicated task. This procedure reduces the need for large, complex projects whenever a new program is required.

The UNIX system not only allows a user the capability to program using UNIX itself, but it also supports several other programming languages. As an example, SYMMETRIX includes interpreters and compilers to develop and run programs in such languages as Pascal, C, Fortran, BASIC, LISP, ICON, and APL. In addition, because UNIX is written in C (a high-level language), system programmers can change the operating system to adapt to their particular needs.

UNIX also supports electronic communications. It contains an electronic mail system which can be used between users on the same system or to communicate with other users

on a remote system. It also supports electronic file transfer, remote file sharing and remote processing.

## C. LIMITATIONS

As in any operating system environment, the use of UNIX presents limitations. Although UNIX was originally developed twenty years ago, many revisions and additions have occurred, and continue to occur, along the way. This updating process has allowed UNIX to correct some past problems and to avoid others. With the many versions of UNIX available, limitations may exist in one version yet not in another. The limitations presented here are general ones which characterize the system as a whole.

### 1. User Interaction

UNIX is often referred to as "unfriendly." However, as the experience and knowledge of users vary, so does the perception of the UNIX system´s "friendliness." A new user of UNIX, especially one completely new to the world of computers, will find that learning the system requires a good deal of time and thought. A lack of system responses or a lack of thoroughly explained error messages, may make the interface with UNIX very difficult. However, to one who is experienced with the UNIX environment, its simplicity and flexibility can be very desirable.

### 2. Processing Environment

Just as the system´s processing environment presents many attractive capabilities, so does it also present some limitations. Technology has advanced greatly since the time when UNIX was initially developed. Because of this, UNIX is based on old concepts. Although new versions of UNIX include some current technological advances, some weaknesses still exist. One of the largest handicaps is that it was not developed for real-time applications and, despite some attempts to modify it, UNIX remains poorly suited for such processing. Also, because of its large size, UNIX is not a good choice for small, single user microcomputers. [Ref. 5:pp. 7-8]

### 3. Standardization

Again, what can be considered advantages of UNIX can also be regarded as liabilities. UNIX was initially written by a small group of people; but, due to its widespread use (especially in the educational environment), many changes and modifications have been made to the system. Because programmers can modify UNIX to their own environment, a lack of standardization has been an ongoing problem with UNIX. The problem has been identified and work is underway to develop a single standard; however, a struggle exists in the industry as to whose version will be the UNIX standard [Ref. 6]. The POSIX system, which is currently being developed,

is an attempt to provide a standard portable operating system based on UNIX.

Another standardization problem for UNIX is its software format. While operating systems for microcomputers generally use standard floppy disk formats, UNIX does not have such a standard. With different hardware manufacturers developing a variety of hardware for UNIX system operations, the format for media storage is also diverse. This is a problem not only for the user who wants to move data from one stand alone system to another, but it is also a major problem for retail software developers. The companies would have to market each of their products in a variety of formats. [Ref. 6] For this reason, UNIX itself does not run any of the PC-based retail programs which have become so popular. These programs include spreadsheet and word processing capabilities.

# III. CONCLUSIONS AND RECOMMENDATIONS

## A. CONCLUSIONS

The use of UNIX has grown tremendously since its beginnings twenty years ago. Its widespread use in the university environment, its simplicity and its capabilities have made it extremely popular today. Considering the propagation of this powerful operating system, it is important that Information Systems students develop an appreciation for, and a working knowledge of, UNIX.

Students could be introduced to the Symmetric 375 during introductory level Information Systems courses. Providing interested students with a copy of the user´s manual will allow them the opportunity to experiment with a UNIX-based workstation on their own. Exploring the system and its uses will not only prepare users to take advantage of its capabilities, but it can also provide them with an increased understanding of computers and operating systems as a whole.

Both a programming course and an operating systems course are required for the Information Systems track. Students could use the Symmetric 375 system in connection with their programming course for editing and developing files for the various language interpreters and compilers available. Additionally, by interacting with the SYMMETRIX

operating system they can better relate to the concepts and theories presented in the operating systems course.

## B. RECOMMENDATIONS

A Symmetric 375 UNIX workstation has been successfully installed in the Information Systems laboratory. This workstation is equipped with the SYMMETRIX operating system, a version of the Berkeley 4.2 BDS operating system [Ref. 1:p. 15]. To ensure proper and continued operation of this computer system, an effective systems administration program must be established. The following recommendations for the program can be accomplished through the system administrator. Appendix B provides a quick reference chart to assist the administrator.

### 1. Location

The Symmetric 375 system should be installed in an area which provides easy access and sufficient workspace. If this is not available, proper operation will be difficult and the situation may discourage system use. It is also important to ensure that proper physical security is maintained and that the system be protected against dust, high temperature, and high humidity. Improper environmental controls may result in damage to the machine.

### 2. Filesystem Backup

To prevent loss of files from an event such as a system crash, procedures for file backup should be followed [Ref. 1:p. 77]. The contents of the system should be saved

to tape on a regular basis. If files are later disrupted or lost, they can be restored from the backup tapes.

### 3. Userid and Password Administration

To provide adequate protection for the system and its data, certain security procedures should be established. Access to the system should be controlled through the use of userid and password assignment.

Group userids can be assigned according to class enrollment. For example, userid "is2100" can be established for use by students enrolled in the IS2100 class. (Because UNIX is "case-sensitive," as explained in Appendix A, it is recommended that a userid not contain capital letters.) Userids can also be assigned to individuals. However, all userids, whether they are being used by a group or by one person, should have an assigned password.

Userids (such as "root" or "owner") which provide special privileges to the user, should be particularly guarded. Passwords associated with these userids should be changed frequently and should be protected from unauthorized use. The superuser privilege "su" which permits unrestricted access to system files, should also be protected in this manner.

### 4. Management of Disk Space

Because the Information Systems laboratory's Symmetric 375 workstation does not have any external storage, it is vitally important that the file space

availability on the hard disk be properly managed. Disk usage can be monitored using utilities provided by the operating system. Resource control, by limiting the number of files or amount of disk space allowed for each user, can also be enforced.

If available disk space is diminishing to a level below what is sufficient for normal operations, files must be removed or compressed. Using the UNIX mail system, directions can be sent to users to remove their files which are no longer needed. This can also be managed with the assistance of instructors whose classes have been assigned group userids. Another option available would be to compress, using the "pack" utility, large system files which are rarely used. Files which are packed cannot be accessed by UNIX programs; therefore, the "unpack" utility must be executed before using the file.

SYMMETRIX offers a utility called "quota" which allows an administrator the ability to ration disk space. He or she can set limits on the number of files belonging to a user or the amount of disk space used. Because this facility requires considerable administration, it is recommended that "quota" not be used unless exhaustion of disk space becomes a severe and frequent problem. [Ref. 1:p. 94]

### 5. References

Users should be provided with a copy of Appendix A, The Symmetrix 375 User's Manual. Additionally, it would be beneficial if other general references on UNIX be maintained in the Information Systems laboratory for student use. Appendix A lists some references which could be considered. Refer to Appendix B for references which can assist the system administrator.

### 6. User Assistance

A point-of-contact, preferably the system administrator, should be provided to the users of the Symmetric 375 system. This is necessary so that the administrator can be notified and can take the proper actions in the event of a system crash or other hardware or software error.

### 7. Maintenance

Because there is no maintenance contract for the Symmetric 375 hardware, it is recommended that the system administrator become familiar with the troubleshooting section and the error indications and diagnostics section of The Symmetric 375 Owner's Manual. Some maintenance and recovery procedures are covered in the manual and may make it possible to avoid outside assistance for minor system repair.

APPENDIX A

NAVAL POSTGRADUATE SCHOOL

SYMMETRIC 375 UNIX WORKSTATION

USER´S MANUAL

TABLE OF CONTENTS

# 1. INTRODUCTION

## 1.1 BACKGROUND

This user´s manual has been prepared for use with the Naval Postgraduate School´s Symmetric 375 UNIX Workstation in the Information Systems´ laboratory. The workstation is a series 32000 microcomputer running the SYMMETRIX operating system, a version of the Berkeley 4.2 BSD UNIX operating system.

Several versions of the UNIX operating system are available today; however, the two major ones are AT&T´s System V and Berkeley´s 4 BSD. Although the same UNIX philosophy is present in both versions, many incompatibilities exist between the two. Berkeley´s 4 BSD UNIX, from the University of California at Berkeley, has become very popular due to its technical improvements over the System V [Ref. 1:p. 6]. This is the version available in the SYMMETRIX software environment.

## 1.2 USER MANUAL CONVENTIONS

No prior knowledge of the UNIX operating system is necessary to use this manual. It has been developed to provide the user with an introduction to the UNIX file system, its commands, and its editing routines.

**Bold** face type is used to depict actual commands the user can enter at the terminal. For example:

**pwd**

is a command that the user enters to display the name of his working directory; more on this command is explained in Section 5.2. It is important to note that all commands must be followed by a carriage return (<**cr**>). This is accomplished by pressing the "return" key on the Symmetric keyboard. The system will not respond to your command until you have hit the <**cr**> to indicate a request to the system. The UNIX system responses are also presented with an indented format but are shown in normal type face. An example of this is:

/usr/is2100

Special keys discussed in the manual include the escape key (<**esc**>), which is in the upper left-hand corner of the keyboard, and the space bar, located at the bottom of the keyboard. One other very important key is the control key. Pressed by itself, the control key has no function. However, when used in conjunction with other keys, it can perform many operations. Some of these operations are discussed in the next section.

## 1.3 TYPING CORRECTIONS

If you discover that you have made a typing error before you press the carriage return <**cr**>, you can make corrections to the line using control characters. For example, when the control key and the letter "h" are pressed at the same time (represented: control-h or ^**H**) the system

will move the cursor back one space and erase any character there. (The backspace key, shown on the keyboard as "**Bs**", does the same function.) Although the "h" is depicted as a capital letter, it is not necessary to hold down the shift key when typing **^H**. The system automatically displays the letter in the uppercase when the control key is pressed.

While the **^H** is used to erase back one character, **^W** allows you to erase back one full word. However, neither of these commands will permit you to correct any further back than the line your cursor is currently on. If you realize that you have made a mistake and wish to erase your entire current line, you can use either **^C** or **^U**. But this must be done before you press the carriage return **<cr>**. Once a **<cr>** has been issued, the line has been presented to the system and can no longer be changed.

## 1.4 UNIX´S CASE SENSITIVITY

The UNIX operating system is what you might refer to as "case sensitive" when dealing with alphabetic characters. It distinguishes between uppercase and lowercase letters. UNIX does not consider "A" equal to "a." Thus, it is very important to ensure that all command words are typed in lowercase, as appropriate. You will most likely find it easier to maintain a file naming convention of using only lowercase letters.

## 1.5 SECURITY

As with any information processing system, security of the resources is vital. It is very important that you take any steps necessary to ensure security of the hardware, software, and the data.

Essential procedures to follow include:

- Do not leave an active terminal unattended.
- Ensure that you have properly logged off before powering down the equipment.
- Do not change the password on any group userid you have access to. Do not disclose a password to unauthorized users.
- Do not consume food or drink at the terminal.
- Report any suspected computer misuse or abuse to the system administrator.

# 2. SYSTEM START UP AND LOG IN PROCEDURES

## 2.1 STARTING THE SYMMETRIC 375

The first step in starting the 375 is to turn on the power to the system and to the monitor. To turn on the system, push the power switch (located on your right side on the back of the 375) up to the on position. After you have completed this, press the button on the front of the monitor to turn it on. The 375 automatically runs internal diagnostics and begins the boot sequence. The autoboot is complete when the following sequence appears:

```
4.2 SYMMETRIX (NPS-is)
login:
```

This should take approximately 4 minutes. The system is now ready for you to log in.

## 2.2 LOGGING INTO UNIX

When the "login" prompt appears, type in the personal or group userid which has been assigned to you and press <cr>. For example:

login: **is2100**

If your userid requires a password, the system will respond with:

```
Password:
```

Go ahead and type in your password followed by a <cr>. Your password will not show on the screen as you type it in.

As explained in Section 1.4, ensure that you use the proper case when entering information. UNIX is case sensitive and does not treat an "A" as equal to an "a".

If you do not enter a valid userid/password combination, the 375 will begin the login sequence again by responding with:

```
Login incorrect
login:
```

If your login procedure is successful, the system will give you the date and time when someone last logged in using that userid and will then return:

```
TERM = (vt220)
```

This is to determine the type of terminal which you are using. Since your Esprit Opus 220 is a vt220 style terminal, simply hit the carriage return <cr>. The system will respond with:

```
%
```

The "%" is the UNIX shell prompt and lets you know that you are in the shell command mode; the 375 is now ready for your commands. If desired, refer to Section 4.2 to gain assistance using the on-line tutorial.

### 3. LOG OUT AND SHUT DOWN PROCEDURES

**NEVER POWER OFF THE SYMMETRIC 375 WITHOUT FIRST LOGGING OUT.** You may log out of UNIX anytime you have the "%" prompt. Simply type:

**logout**

followed by a <cr>.

When you have successfully logged out, the system will respond with:

4.2 SYMMETRIX (NPS-is)

login:

At this point, the log in sequence can be executed or the system can be shut down. Turn off both the monitor and the system using the button and switch which you used to turn them on.

# 4. ON-LINE REFERENCE AND TUTORIAL

## 4.1 ON-LINE REFERENCE

Like most UNIX systems, the 375 has an on-line reference manual. To consult the on-line manual, type **man** followed by a command name. For example:

**man who**

will give you a full description of the **who** command. Typing:

**man man**

will describe the **man** command including the format for using it. **man** displays its results using a program called **more**. This allows one screen of information to be displayed at a time. In the bottom left-hand corner of your screen is the word "more" followed by a percentage. This percentage shows how much of the **man** entry has been displayed so far. By pressing the space bar, you can advance to the next screen or, by pressing the **<cr>** , you can advance just one line. When you reach the end of the entry, the shell prompt will appear. If you desire to exit before you reach the end of the entry, type control-c (**^C**). This will return you to the shell prompt immediately.

## 4.2 ON-LINE TUTORIAL

SYMMETRIX offers an on-line computer aided instruction course. It´s a beginner´s course on UNIX which covers the following subjects:

- files: basic file handling commands
- editor: text editor
- vi: text editor (screen-oriented)
- morefiles: more information on file manipulation
- macros: text formatting commands (for nroff)
- equ: used when typing mathematical equations
- C: writing programs using "C"

To start the course, simply type:

**learn**

and, as with any command, follow this with a <**cr**>.

Each subject has several lessons and you will have the choice to cover the subjects in any order which you desire. However, if you are a beginner to UNIX, it is recommended that you complete the subjects in the order given. You may stop at anytime in a lesson by typing:

**bye**

Remember the last lesson which you complete so that, when you return to the course, you can begin where you left off.

Because the **learn** course is based on standard Unix, you may find some differences between it and this manual. The manual was written specifically for the SYMMETRIX software environment, rely on it when there are discrepancies between the two. To learn more about the on-line tutorial **learn**, type:

**man learn**

and read the manual´s reference on the command.

# 5. FILE MANAGEMENT

## 5.1 INTRODUCTION TO THE FILE SYSTEM

One of the best features of UNIX is its simple filesystem. It is made up of the following:

- Directories. Collections of files and other directories. They provide a means to logically group files.
- Ordinary files. A collection of characters; such as text files, programs, or data.
- Special files. Files which represent input/output devices such as a printer or terminal.

## 5.2 DIRECTORIES

Using directories, the filesystem is organized in a hierarchical structure - like a tree. The highest level directory is the "root" directory. It is represented by a slash (/). There are many second level directories on the system but the one with which you should be most concerned is the "/usr" directory; your login directory will be under it. A sample hierarchical directory is represented in figure 1.

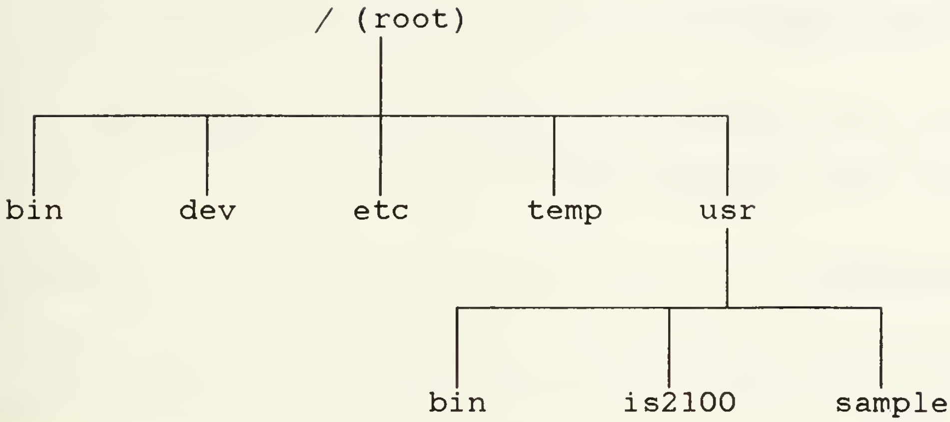


Figure 1. A UNIX Hierarchical Directory

When you log in, the system will automatically put you into your login or "home" directory. This is the directory which will contain the files of everyone who logs in using that userid. To find out the pathname of your login directory, type:

**pwd**

The command **pwd** stands for "print working directory." You can type this command at anytime to find out which directory you are currently in. For example, if you logged in with the userid "is2100", the system would respond to your **pwd** command with:

/usr/is2100

To move into another directory, use the **cd** (change directory) command. Type **cd** followed by the pathname of the directory which you desire. For example:

**cd /usr/sample**

moves you to the "sample" directory. If you type **pwd** now, the system would respond with:

/usr/sample

To return to your home directory at any time, simply type:

**cd**

with no pathname. It is important to ensure that you are in your own login directory when creating files.

## 5.3 CONTENTS OF A DIRECTORY

To discover the contents of a directory, you can use the **ls** command. Type:

**ls**

This will list the names of all the files and directories in your current directory in alphabetical order. To view the contents of another directory, you can change directories using the **cd** command and then use **ls** (with no arguments); or, you can stay in you current directory and type **ls** followed by the desired pathname. For example:

**ls /usr/sample**

will list all the directories and files under the /usr/sample directory.

The **ls** command has several options available. For example:

**ls -l**

will display more information about the files in a directory such as its size and the date and time it was last modified. To learn more about the options available for **ls**, type:

**man ls**

to review the manual entry. (If you are unfamiliar with the on-line manual, refer to Section 4.1.)

## 5.4 DISPLAYING A FILE

Once you know the name of a file and its pathname, you can examine its contents. First, ensure that you are in the same directory as the file you wish to display. Do this by using the **pwd**, **ls**, and **cd** commands as necessary. Then use either **more** or **cat**, as explained below, to view the file.

The **more** command is often convenient for displaying a file which is more than one screen long. Type **more** and the name of a file which is in the current directory. To demonstrate this, go to the sample directory by typing:

**cd /usr/sample**

then type:

**more file.one**

The first screen of the file is displayed. As explained in Section 4.1, you can advance to the next screen by pressing the space bar. The percentage which follows the word "more" in the lower left-hand corner of the screen is the percentage of the file which has been displayed so far. You can also advance the file one line at a time by pressing the return key **<cr>**.

Another way to display a file is with the **cat** command. **cat** uses the same format as **more** (command followed by the filename). For example:

**cat file.one**

will display the contents of file.one if a file with that name exists in the current directory. Otherwise, you will be given an error such as:

No such file or directory.

When **cat** displays a file it does not display one screen at a time. It will print out the contents of the whole file before pausing. To suspend the output on the screen so that it does not scroll past too quickly, type control-s (^**S**) when you want the scrolling to stop. To resume scrolling, type control-q (^Q). As you can see, it is often easier to view a longer file using **more** rather than **cat**.

## 5.5. COPYING A FILE

To make a duplicate copy of a file, use the copy command (**cp**). This command copies the contents of one file into another. For example, to make a copy of a file named "memo" and name it "backup.memo", type:

**cp memo backup.memo**

You can then use the **ls** command to verify that the file copy was made. Be careful when choosing a name for the new copy. If a file with that name already exists in your directory, it will be written over when you execute the **cp** command. The system does not provide you with any warning to let you know that a file by that name already exists.

## 5.6. RENAMING OR MOVING A FILE

The move command (**mv**) allows you to rename a file in your directory or move a file from one directory to another. To rename a file, type **mv** with the old filename followed by the new name. For example:

**mv old.name new.name**

will rename the file "old.name" to "new.name."

The following are guidelines to consider when naming or renaming a file:

- a name can consist of 1 to 14 characters.

- all characters other than "/" are legal. However, it is best to use only letters, numbers, the period (.), and the underscore (_).
- do not use a period (.) as the first character in a filename.
- UNIX distinguishes between uppercase and lowercase characters. Therefore, for typing ease, you should not use both cases in a single filename.
- use meaningful names which identify the contents of the file.

If you create a file under another directory and then want to move it to your own directory, you will also use the **mv** command. First you must know the pathname for the directory to which you want to move the file. Then you must ensure you are in the directory that has the file you would like to move. You can do an **ls** to check this. If you are not in the proper directory, use the **cd** command (explained in Section 5.2) to change directories. To move the file to another directory without changing the name of the file, type **mv** followed by the pathname of the directory to which you want to move the file. To move "file.one" to a directory with the path "/user/is2100", type:

**mv file.one /user/is2100**

However, if you would like the name of the file changed to "file.two" when you move it. Type:

**mv file.one /user/is2100/file.two**

You can verify that the **mv** command worked by using the **ls** command to list the contents of the directory.

## 5.7 REMOVING A FILE

The remove command (**rm**) allows you to delete a file. To remove a file, simply type **rm** and then the name of the file which you wish to delete. For example:

**rm old.file**

will delete "old.file." CAUTION. ONLY DELETE FILES WHICH YOU HAVE CREATED YOURSELF. DO NOT USE THE **RM** COMMAND ON ANY OTHER FILES.

The system will not print a message confirming that the file was deleted; however, you may verify it by using the **ls** command.

The **rm** command is a very valuable command. It is easy to use and it allows you to keep the system clear of your excess files. But, be careful to ensure that you are using the command properly; UNIX does not give you a chance to verify your action before it removes the file.

# 6. TEXT EDITING

## 6.1 INTRODUCTION

The 375 supports two text editors: **ed**, a line editor, and **vi**, a screen editor. The editors can be used to create new files and to change existing ones. Both **ed** and **vi** place a copy of your file in a temporary storage area (a buffer) so that, when you are making changes, you are actually changing the copy and not the original file. Before exiting the editors, you can save the text in the buffer to your file or, if you decide that you do not want any of the changes you made, you have not corrupted your original file. For security purposes, do not use either editor to change any file which you did not create yourself.

## 6.2 THE ed LINE EDITOR

**ed** is a text editor that allows you to create text files and to edit them one line at a time. Since full screen editing is not possible with **ed**, the editing of a even a few lines using the "line-by-line" method can be very time consuming and confusing. It is highly recommended that you use a full screen editor, such as **vi**, for any significant editing. However, if you undertake an editing process for which a line editor is sufficient, here are some basics on **ed**.

The editor operates in two modes: the command mode and the text input mode. When you initiate the editor, you will

be placed in the command mode. In this mode, you can do things such as: add, change, and delete text, save your file, and exit the editor. The **ed** editor assigns line numbers to each line in the buffer to allow you to address and identify specific lines. You can make corrections to each line by following the same procedures used when correcting a line of type in the UNIX shell. (These procedures are discussed in Section 1.3.)

To begin an **ed** editing session, type **ed** and the name of the file. For example:

**ed myfile**

If "myfile" is a new file, the system will return a "?" followed by the filename. For example:

? myfile

However, if "myfile" already exists, the system will respond with the number of characters currently in the file. Since you are now in the command mode, **ed** is ready to except your command. As with shell commands in UNIX, **ed** commands are case-sensitive. Ensure that you use uppercase or lowercase characters, as appropriate. The following is a list of some basic **ed** commands and their meanings.

**ed** Commands

| | |
|---|---|
| **a** | append text after the current line (puts you in the input mode). |

**i** insert text before the current line (puts you in the input mode).

**.** end input mode and return to command mode. This must be typed as the first character on a line by itself.

**p** display lines of the text. Examples: **1p** will display the first line; **3,4p** will display lines 3 and 4; **2,7p** will display lines 2 through 7; **1,$p** will display the entire text. ("**$**" is the last line symbol.)

**c** change lines of text. Examples: **5c** will change line 5; **2,9c** will change lines 2 through 9; **c** will change the current line of text.

**d** delete lines of text. The line number format is the same as it is for **p** and **c**, ie. **4d** will delete line 4.

**m** move line(s) after another line. Examples: **5m10** moves line 5 after line 10; **3,6m27** moves lines 3 through 6 to follow line 27.

**r** reads in a file to add to the text. The format for the **r** command is: line number of text to follow (default is current line), **r**, and then the filename. Examples: **10r prices** adds the data from file "prices" to the text file following line 10, **r info** inserts the data from file "info" following the current line.

**w** write to file. Default is the current file (the one named in the **ed** command line) but another file can be specified. For example, **w myfile** will save what is in the buffer to a file named "myfile."

**q** quit **ed**. If you have made changes to the buffer without saving them, **ed** will print a "?" to remind you to save the text before quitting.

**Q** quit **ed**. This does the same as **q** but does not warn you if you have not saved changes.

---

Additional information on **ed** and its commands is available in the on-line manual; type:

**man ed**

## 6.3 THE **vi** SCREEN EDITOR

The **vi** editor, like the **ed** editor, allows you to create, change and save text; however, **vi** can display an entire screen of the file at one time and let you edit anywhere on the screen by allowing you to scroll the text up and down. You also have control over the cursor and can move it anywhere on a line or anywhere in the file to add or change text. This can be very convenient. The editor operates in two modes: the input mode and the command mode. When you initiate the editor, you will be placed in the command mode but, you will notice, because you can move the cursor around the screen, there are no line numbers for addresses. Starting a **vi** session is very similar to starting **ed**, just type **vi** and the name of the file then press <**cr**>. For example:

**vi myfile**

If "myfile" already exists, **vi** will print a line on the bottom of the screen showing the name and size (in number lines and number of characters) of the file. It will also use the remainder of the screen to display the file. If a file is more than one screen long, only the beginning of the file will be displayed.

If "myfile" is a new file, the bottom line on the screen will show the filename and then "[New file]." The

other lines below the cursor will have tildes (~). This is how **vi** indicates that there is no text.

Since you are now in the command mode, **vi** is ready to except your command. As with shell commands in UNIX, **vi** commands are case-sensitive. Ensure that you use uppercase or lowercase characters, as appropriate.

To start building a new file, you will want to issue the command to insert text (**i**). Just type:

**i** (with no <**cr**>)

You will notice that nothing appears to have happened. This is alright; **vi** does not display most commands when you type them and does not indicate when you change modes. You also do not press carriage return <**cr**> after issuing a **vi** command. You are now in the input mode and can begin entering text.

If you make errors while typing in the text, you can correct some now or wait until you are editing. It is recommended that you follow the same procedures used when correcting a line of type in the UNIX shell (discussed in Section 1.3) only when the error is a few spaces from the cursor. All other errors you should correct while editing the file.

When you have completed adding text, press the escape key <**esc**> to return to the command mode. At this level, you

can decide to go back and edit the file, save the file, or exit **vi**.

The following is a list of some basic **vi** commands and their meanings.

**vi** Commands

---

Moving The Cursor:
(These commands are not followed by a <**cr**>.)

- **h** move to the left.
- **j** move down.
- **k** move up.
- **l** move to the right.
- **^w** move forward one word.
- **^b** move back one word.
- **^D** move down half a screen.
- **^U** move up half a screen.
- **^F** move forward a full screen.
- **^B** move back a full screen.

Entering text:
(These commands are not followed by a <**cr**>.)

- **i** insert text.
- **a** append text (after the current cursor position).
- **0** (zero) enter text at beginning of current line.
- **o** enter text after current line.
- <**esc**> exit from input mode and return to command mode.

Deleting text :
(These commands are not followed by a <**cr**>.)

**x** delete the character at the cursor.

**r** delete the character and replace with the next letter typed.

**R** delete and replace more than one character (typeover).

**dw** delete the word at the cursor.

**db** delete the preceding word (one word back).

**dd** delete the line.

Saving the file and Exiting **vi**:
(Press <**cr**> after issuing one of these commands.)

**:w** save to file. (Default is file named in **vi** command line. To save to another file, follow **:w** with the appropriate filename.)

**:r** read contents of a file to the workspace. (**:r** must be followed by the filename.)

**ZZ** exit **vi** and save file.

**:q** exit without saving file. **vi** will issue a warning if you have made changes to the file and have not saved it. Type **:quit!** to override this warning.

**:q!** exit without saving file and getting no warning.

---

Additional information on **vi** and its commands is available in the on-line manual; type:

**man vi**

# 7. UNIX SHELL AND COMMANDS

## 7.1 THE UNIX SHELL

The shell is your user interface to the UNIX system. It is a powerful program that interprets and then executes your typed commands. The shell can also be used as a programming language (refer to Section 11.6 for suggested references to explain its use as such).

There are several different shell programs available for the UNIX operating system. The two most popular and widely used ones are the Bourne shell and the C shell. Both programs are very similar although the C shell does contain some features which are not available in the Bourne shell. [Ref. 2:p. 10] This user´s manual will deal specifically with the C shell and will introduce some of the functions you can perform within the shell.

## 7.2 THE SHELL PROMPT

When you log into SYMMETRIX, you are in the C shell and are given the "%" prompt. In Section 2.2, the "%" sign is referred to as the UNIX shell prompt. This is because it indicates that you are in the shell program and can enter shell commands.

## 7.3 SHELL COMMAND FORMAT

There are three parts to a shell command: the command name, the option(s), and the argument(s). This is often depicted as follows:

**command [options] [arguments]**

Options and arguments are shown in brackets because, depending on the command and its use, they may not be required.

The command name is simply the name of the task you want to execute. For example,

**ls**

is the command which will list the contents (the names of the files and directories) in the current directory. Options modify how a command works while arguments name where it is to work. To demonstrate the use of options and arguments, we could type:

**ls -l /usr/sample**

The option -l will display more specific information about the files and directories and the argument **/usr/sample** names the directory whose contents we wish to list.

The proper format of a specific command, its options, and arguments can be found in the command´s manual (**man**) entry. Refer to Section 4.1.

## 7.4 INPUT/OUTPUT REDIRECTION

The standard input and output device for the 375 is the terminal. Unless otherwise specified, when you execute a program the system will take information from the terminal

as input and will display any output back to the terminal. One very convenient feature of UNIX, input/output redirection, allows you to designate your input and output devices.

The "less than" sign (<) permits you to define the input for your program. Using the **mail** program, here is an example:

**mail janedoe < memo**

This command will mail the contents of the file "memo" to user "janedoe." The < instructs the **mail** program to take the contents of "memo" as its standard input.

To redirect the output from a program, the "greater than" sign (>) can be used. For instance, if you wish to create a file named "mylist" which contains the contents of your current directory, type:

**ls > mylist**

The list of files and directories will not be displayed on the terminal screen but will instead be written to a file named "mylist." If "mylist" already exists before you redirect this output to it, anything in the file will be overwritten and lost.

Sometimes a file exists which you would like to add to but not overwrite. The double "greater than" sign (>>) can

be used to redirect output and accomplish this. For example:

```
ls /usr/sample >> mylist
```

will append "mylist" with the contents of the directory "/usr/sample." Therefore, the file would now contain the contents of both your current directory and the "/usr/sample" directory.

## 7.5 PIPES

Another way to control the standard input or output of a program is by using a "pipe" which is represented by a vertical bar (|). A pipe is used to pass the output from one program into another program as the standard input.

To count the number of files and directories listed in the current directory, you could use the **ls** (list) command and the **wc** (word count) command as follows:

```
ls > tempfile
wc < tempfile
```

The first line "**ls > tempfile**" will take the contents of the current directory and write it to a file named "tempfile." The line "**wc < tempfile**" redirects "tempfile" (which now contains a list of files and directories) to be used as the input to **wc**. The **wc** program will count the number of words in tempfile and display the answer on the screen.

Using a pipe (|) and the same two commands, you can achieve identical results by typing:

**ls | wc**

This not only saves you from typing two longer command lines, but it also does not require the use of a temporary file to store intermediate results.

If, however, you have a task for which you desire to keep the intermediate results of a pipe process, you can use the **tee** utility in the following manner:

**ls | tee tempfile | wc**

This will save the list of the directory's contents in a file named "tempfile" which can be used later.

7.6 FILENAME GENERATION

Using some special characters called "metacharacters", you can cause the shell to generate filenames which are currently on the system. This shortcut can be very helpful when working with a large number of files with similar names.

The metacharacters "?", "*", and "[]" match a character or string of characters in a filename and can be placed anywhere in the filename. They are used to:

? - match a single character

* - match a string of any length

[] - match any of the characters which are enclosed in the brackets.

Using the **ls** command, the following are examples utilizing the metacharacters.

**ls chapter?**

The ? replaces one character in the filename; therefore, this command will list all files in the current directory which begin with "chapter" and have one more character. For example, the shell might respond with:

```
chapter1
chapter2
chapterz
```

Because the ? represents only one character, it will not list names such as, "chapter1a" or "chapter12." These could, however, be listed by typing:

**ls chapter??**

If you would like to depict multiple characters in a filename, the * can be used. For example:

**ls chapter***

will prompt the shell to list all files whose names start with "chapter", regardless of how many more characters may exist in the name. The following are examples of names that the shell might list:

```
chapter1
chapter5.a
chapter71
chapterindex
```

As with the ? symbol, the [] is used to represent just one character in a filename. However, using these brackets, allows you to designate specific characters (or a range of characters) which can be represented. For example, if you want to see if any files exist named "chapter1", "chapter2", or "chapter3", type:

**ls chapter[123]**

The shell will look for "chapter" followed by a "1", "2", or "3." Because brackets ([]) can only be used to represent and match a single character, the shell will not list a file named "chapter123."

A range of characters can be used within the brackets. Therefore, typing:

**ls chapter[1-3]**

will produce the same results as typing:

**ls chapter[123]**

A range of letters can also be used within the []. For example, if you use the range **[a-z]**, the shell will search for lowercase letters; while, if you use **[A-Z]**, it will search for only uppercase letters.

Once you understand the meaning and use of these three metacharacters, you can gain an even greater advantage by using them together. For example:

**ls [A-Z]***

will provide a list of all files which begin with a capital letter. While:

**ls ?[0-9]**

will list all two character filenames which end with a number.

## 7.7 HISTORY

Another shortcut offered to UNIX users by the C shell is the **history** mechanism. (This capability is not available with the Bourne shell.) The **history** program stores your most recently executed command lines and allows you to reexecute them. To see a list of the recent command lines which have been stored type:

**history**

The shell will return a list of the command lines and their line numbers. This is an example of what you might see:

```
7 % history
    1 pwd
    2 cat myfile
    3 cd /usr/sample
    4 ls -l
    5 more file.one
    6 cd
8 %
```

To reexecute one of the stored lines, type an exclamation point (!) followed by a reference to the desired command line. You can reference the line by the absolute line number, a relative line number or by the contents of the text. The following methods could all be used to reexecute line 2, **cat myfile**.

- To reference the command by the absolute line number, type:

  **!2**

- To reference the command by the relative line number, type:

  **!-6**

  (The current line number is 8 and you wish to reference line 2; therefore, subtract 6 lines.)

- To reference by the text, type:

  **!cat**

  (The shell will search for and execute the most recent event which begins with the string "cat.")

Another option which exists when using the **!** command is to execute the previous command line by typing:

**!!**

This command will always execute the last command line which you previously typed. It produces the same results as typing:

**!-1**

### 7.8 SHELL SCRIPTS

So far in this section you have been introduced to many time saving techniques. This final segment will present one more shell procedure which can demonstrate the convenience of using UNIX.

The **history** mechanism allows you to reexecute previously typed command lines. But, what if you have a series of command lines which you frequently execute in succession? Shell scripts, also referred to as executable shell programs, can provide you with the means to perform several different tasks by typing only one line. The shell script is actually a series of commands which is stored in a file to be executed whenever desired. For instance, if you execute the following commands every time you log on, you can build them into a shell script.

```
cd /usr/sample
ls > samplelist
cd
ls -l > mylist
cat stufftodo*
```

Simply build a file containing these commands. (If you need a review on creating and editing files, refer to Section 6.) To ensure that the file is executed by the C shell, rather than the Bourne shell, the first line of the file must begin with the pound sign character (#). Any text on the line which follows the #, will be viewed as a comment and is not executable. Therefore, any commands you wish to execute should be typed on subsequent lines. The # can also be used elsewhere in the file to add comments. This is a sample script shell:

```
#  This is my start up shell script
cd /usr/sample
ls > samplelist
cd
ls -l > mylist
cat stufftodo*
```

To execute this file, type **sh** and then the filename. For example. if you named the file "start.up" type:

```
sh start.up
```

The shell will execute each of the lines in the script as if you had just typed them. You can even make the start up procedure simpler so that you do not have to type the **sh**. To do so, simply type the following command once:

**chmod +x start.up**

The **chmod +x** command followed by your filename makes the file directly executable so that you only have to type the filename to execute it. Therefore, you will only have to type:

**start.up**

## 7.9 MULTITASKING

The UNIX operating system allows you to execute more than one job at a time. This multitasking can greatly increase your productivity. When you initiate the execution of a job in UNIX, the shell will run the entire process and then issue the shell prompt "%" when it is complete. To run more than one job at a time, you can start a job and have it run in background mode. To start a process in background mode, simply end the command with an ampersand (&). This will instruct the shell to begin execution of the job but to return a shell prompt so that you can continue executing other commands. When you initiate a job in background mode, the shell will return a process identification (PID) number to provide you with a means for identification of the process. This is an example of submitting a process for execution in background mode:

```
% ls -l /usr/sample > mylist &
214
%
```

After you type your command line and hit the carriage return **<cr>**, the shell will respond with the PID (in this case, 214) and then return a shell prompt. You are now free to continue executing other commands.

To check on the status of any process which you have running, you can use the **ps** command. Simply type:

**ps**

The shell will respond with information about any process which you have active. The following type of information will be displayed:

| PID | TTY | TIME | COMMAND |
|---|---|---|---|
| 214 | 1 | 0.01 | ls |

The PID is the process identification number, TTY is the terminal identification, the TIME is the cumulative execution time for the process, and COMMAND is the command name being executed. You can receive even more information by using the **-l** option as follows:

**ps-l**

The **kill** command can be used to end a process before it has finished. The process to be terminated must be identified by its PID. For example:

**kill 214**

If you wish to terminate all processes you have running, you can simply type:

**kill 0**

This command will kill all your processes with the exception of your login shell.

# 8. UNIX UTILITIES

## 8.1 PRINTING A FILE

A line printer is configured to the Symmetric 375 and can be used to print out files from the system. Ensure that you have powered up the printer before sending anything to print and ensure you power it down when you log off the system.

One way to print a file is by redirecting output as explained in Section 7.4. The device filename for the line printer is "/dev/lp." Therefore, to send something to the printer, rather than having it display on the terminal screen, redirect it to this special file. If you want to print out a list of your current directory's contents, type:

**ls > /dev/lp**

Another way to direct output to the printer is with the **lpr** command. This command sends your requested file to the print queue. If the printer is available, the file will print out proceeded by a banner page. If, on the other hand, the printer is already busy, your file will be put in a queue and typed when the printer is free. The **lpr** command does not format or paginate your files before printing them; therefore, it is often wise to format them for printing before sending them to the queue. The **pr** command can be used to format and paginate the file. If you want to format "myfile" for printing, you can do so by typing:

**pr myfile**

However, it is often difficult to review the formatted copy of the file on the screen as it scrolls past. Therefore, the following steps are recommended if you wish to print a file more than one page long. First, format the file with **pr** but direct the output to a file. For example:

**pr myfile > myfile.pr**

You can now review "myfile.pr" with **more** to ensure the file is formatted the way wish to print it. Then go ahead and send it to the printer with the following command:

**lpr myfile.pr**

Ensure you reference the formatted version of the file (the one we added the ".pr" extension to).

Both the **pr** and **lpr** commands have options which can be assigned to them. To get an explanation of these options and their use, consult the on-line manual listing (**man**) for each command. Refer to Section 4.1 for assistance.

## 8.2 SPELL UTILITY

The **spell** utility checks the spelling of words in a file by comparing each word to a dictionary file. To use, type the command followed by the filename. For example, to check the words in "file.one", type:

**spell file.one**

Words which do not match any listings in the dictionary file will be displayed to the screen. If you are using the **spell** utility on a long file or on one in which you expect many misspelled words, you can direct the output of **spell** to a file rather than have it display on the screen. For example:

**spell file.one > misspell**

## 8.3 COMPILERS AND INTERPRETERS

The Symmetric 375 has several resident compilers and interpreters which will be introduced briefly in this section. Each compiler translates a program into executable form. By typing the name of the executable filename, the program can then be run. The interpreters, on the other hand, execute when the commands are types.

The C compiler is **cc**. To run a C program, first create the program using one of the editors available on the system. When the file has been created, compile it using the **cc** command. For instance, if we had a C program named, "progl.c", we would type the following line to compile it:

**cc progl.c**

If no options are used (as in our example), the executable file which is output from the compiler is named "a.out." This is the system default. However, if you would like to

name the executable file "progl.exec", use the **-o** option as follows:

```
cc progl.c -o prog.exec
```

You now have a program file named "prog.exec." To execute it, simply type its name:

```
prog.exec
```

The Pascal compiler is **pc** and is evoked in the same manner as the C compiler. To execute a Pascal program, follow the same procedures explained above for the C program but substitute "**pc**" (the Pascal compiler) for "**cc**" (the C compiler). Also follow the same procedures to execute a Fortran program but substitute "**f77**" (the Fortran compiler).

A LISP interpreter and BASIC interpreter are also available on the 375. To implement an interpreter, simply type the interpreter command ("**lisp**" for LISP and "**basic**" for BASIC) and then type any commands you wish to execute. For example, here is a sample BASIC program:

```
basic
10    print "This is a BASIC program"
run
```

The system will respond with:

```
This is a BASIC program
```

If you have any questions on the interpreters or compilers, consult the **man** entries for more information.

# 9. MAIL

## 9.1 HOW TO SEND MAIL

You can send mail to any user as long as you know their userid. You can also send mail to yourself or others who log on using the same userid as you. To send mail, type:

**mail** userid

For example, to send mail to someone with userid "johndoe", simply type:

**mail johndoe**

After you press the return, **<cr>**, you are in the **mail** program. Anything which you type from now until you exit the program is part of your message. If you make any errors while typing, you can make corrections only on your current line. Make these corrections by following the same procedures used when correcting a line of type in the UNIX shell. These procedures are discussed in Section 1.3. If you want to cancel the message you are typing, press control-c (**^C**) twice. The message will be cancelled and you will be returned to the shell prompt.

When you have finished typing your message, advance to a new line and type a period (.) at the beginning of this line. Then press **<cr>**. Another way to do this is by pressing **^D** at the beginning of a new line. The system will return a shell prompt.

To mail the same message to more than one user, list each userid separated by a space. If you want to send the message to user johndoe and user janedoe, type:

**mail johndoe janedoe**

If you have a longer message to mail, you may want to prepare it using a text editor. After creating and editing the message, store it in a file. You can then direct this file into the mail system and send the message to another user. For example:

**mail janedoe < memo**

will mail the contents of the file "memo" to user "janedoe." The "less than" sign (<) instructs the **mail** program to take the contents of "memo" as its standard input.

## 9.2 HOW TO READ MAIL YOU RECEIVE

When you log into the system you may receive the message "You have mail." You can receive mail from other users or from the system itself. To view the mail sent to you, type:

**mail**

with no arguments. The system will display a message prompting you to choose an option at the "&" prompt. If you need help, type:

?

This will give you a list of all the options which you can execute. Some options include:

| | |
|---|---|
| **n** | display the next message |
| **<cr>** | the current message will not be deleted and the next message, if there is one, will be displayed. |
| **d** | delete the current message and go on to the next. |
| **p** | display the message again. |
| **s** | save the message to a file called "mbox." To save to a different file, include the filename following the **s** option. Example: **s msg**, will save the message in a file called "msg." |
| **q** | quit the mail program. (^D will also do the same). |

To learn more about the **mail** program and its other options, look at the manual entry by typing:

**man mail**

# 10. PROBLEMS AND ASSISTANCE

## 10.1 PROBLEMS DURING SYSTEM BOOT

If, after following the system start up procedures in Section 2.1, you are unable to get the "login" prompt, power down the system and begin the procedures again. If the system fails to properly boot on the second try, the system should be reset. The reset button is on your left side on the back of the Symmetric 375 system unit. Press this button <u>once</u>. Allow the system approximately five to six minutes to finish the reset and boot sequence. If the system is still unsuccessful in booting, power down the equipment and report the problem to the system administrator.

## 10.2 LOCKED KEYBOARD

If while using the Symmetric 375, you experience a problem where nothing occurs when you attempt to type in commands or press keys, the keyboard may be locked. This might be indicated by the word "HOLD" in the lower left-hand corner of the screen. To eradicate this problem, type control-q (^Q). This is accomplished by holding down the control key while pressing the letter "q."

This procedure should free the keyboard and allow you to begin entering commands. If, however, it is unsuccessful, follow the reboot procedures presented in Section 10.1.

## 10.3 SYSTEM CRASH

If a system crash occurs while you are operating the Symmetric 375, power down the equipment and begin start up procures. If the procedures fail to recover the system, power down the equipment and report it to the administrator. If you are successful in rebooting the system, you may receive mail from the root telling you how to recover any files which may have be disrupted during the crash. If you were editing a file when the system went down, you can usually retrieve most of your file changes with the editor´s "recover" option. For instance, if you were editing a file named "test.one" in **vi** at the time of the system failure, you can type the following command line to attempt recovery of any changes you made to the file:

**vi -r test.one**

# 11. UNIX MANUALS AND REFERENCES

For an overview on UNIX, including such topics as its history, its philosophy and its practical applications, consider the following references:

An Introduction to Operating Systems by Harvey M. Deitel, Addison-Wesley Publishing Company, Inc., 1984.

The UNIX Environment by A. N. Walker, John Wiley & Sons, Inc., 1984.

The UNIX Operating System by Kaare Christian, John Wiley & Sons, Inc., 1988.

UNIX Internals: A Systems Operations Handbook by Myril Clement Shaw and Susan Soltis Shaw, TAB BOOKS, Inc., 1987.

The hardcopy of the online reference manual is published as:

UNIX Programmer's Manual, Volume 1 from Bell Telephone Laboratories, Inc., 1983.

Information on UNIX commands and the file system is included in the these manuals:

A Practical Guide to the UNIX System, second edition by Mark G. Sobell, The Benjamin/Cummings Publishing Company, Inc., 1989.

The UNIX for Beginners Book: A Step-By-Step Introduction by Bryan Strong and Jay Hosler, John Wiley & Sons, Inc., 1987.

The UNIX System User's Guide from AT&T Bell Laboratories, Prentice-Hall, 1986.

Understanding UNIX: A Conceptual Guide by James R. Groff and Paul N. Weinberg, Que Corporation, 1988.

To learn more about editing in UNIX, see:

Editing in a UNIX Environment: The vi/ex Editor by Mohamed el Lozy, Prentice-Hall, Inc., 1985.

Quick reference guides to UNIX commands include:

UNIX RefGuide by McNulty Development Inc., Prentice Hall, 1986.

UNIX System Command Summary for Berkeley 4.2 & 4.3 BSD by Specialized Systems Consultants.

The following manuals provide guidance for programming on the UNIX system:

The UNIX Programming Environment by Brian W. Kernighan and Rob Pike, Prentice- Hall, Inc., 1984.

UNIX and XENIX: A Step-By-Step Guide by Douglas W. Topham and Hai Van Truong, Brady Communications Company, Inc., 1985.

UNIX Programmer's Manual, Volume 2 by Bell Telephone Laboratories, Inc., 1983.

APPENDIX B

SYSTEM ADMINISTRATOR RESOURCES GUIDE

1. To assist the Symmetric 375 Systems administrator, references are provided for the following procedures:

| Procedure | Reference |
|---|---|
| Access to superuser or root (su) | see [Ref. 1:p. 72] |
| Accounting (cron) | see [Ref. 1:p. 93] |
| Adding and deleting users (nu) | see [Ref. 1:p. 71] |
| Changing the date and time (date) | see [Ref. 1:p. 75] |
| Creating directories (mkdir) | see [Ref. 8:p. 96] |
| Error indications and diagnostics | see [Ref. 1:p. 63] |
| Exhaustion of disk space | see [Ref. 1:p. 85] |
| File mode change (chmod) | see [Ref. 8:p. 193] |
| Filesystem backup | see [Ref. 1:p. 77] |
| Monitoring system performance (systat) | see [Ref. 1:p. 93] |
| Password assignment (passwd) | see [Ref. 8:p. 106] |
| Reading and writing a tape (tar) | see [Ref. 1:p. 35] |
| Remove directory (rmdir) | see [Ref. 8:p. 127] |
| Resource control (quota) | see [Ref. 1:p. 94] |
| Restoring a filesystem | see [Ref. 1:p. 83] |
| System and file recovery (fsck) | see [Ref. 1:p. 97] |
| Tape dump (dump) | see [Ref. 1:p. 80] |
| Trouble shooting | see [Ref. 1:p. 37] |

# LIST OF REFERENCES

1. Symmetric Computer Systems Corp., The Symmetric 375 Owner´s Manual, 1985.

2. Topham, D. W., and Troung, H. V., UNIX and XENIX: A Step-By-Step Guide, Brady Communications Company, Inc., 1985.

3. Silberschatz, A., and Peterson, J. L., Operating System Concepts, alternate ed., Addison-Wesley Publishing Co., 1988.

4. AT&T Bell Laboratories, The UNIX System User´s Guide, Prentice-Hall, 1986.

5. Shaw, M. C., and Shaw, S. S., UNIX Internals: A Systems Operations Handbook, Tab Books, Inc., 1987.

6. O´Connor, R. J., "The Long-awaited Age of Unix is Finally Beginning to Dawn," The San Jose Mercury News, v. 138, p. 1F, March 5, 1989.

7. Christian, K., The UNIX Operating System, 2d ed.,John Wiley & Sons, Inc., 1988.

8. Bell Telephone Laboratories, Inc., UNIX Programmer´s Manual, 7th ed., v. 1, 1983.

## INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Defense Technical Information Center<br>Cameron Station<br>Alexandria, VA 22304-6145 | 2 |
| 2. | Library, Code 0142<br>Naval Postgraduate School<br>Monterey, CA 93943-5002 | 2 |
| 3. | Dr. Norman F. Schneidewind, Code 54Ss<br>Department of Administrative Sciences<br>Naval Postgraduate School<br>Monterey, CA 93943-5000 | 2 |
| 4. | Dr. Magdi Kamel, Code 54Ka<br>Department of Administrative Sciences<br>Naval Postgraduate School<br>Monterey, CA 93943-5000 | 2 |
| 5. | Curricular Officer, Code 37<br>Computer Technology Programs<br>Naval Postgraduate School<br>Monterey, CA 93943-5000 | 1 |
| 6. | LT Charlotte V. Smith<br>NMPC - 471<br>Naval Military Personnel Command<br>Washington, D.C. 20370 | 2 |